Patch-based inpainting detection

Antoine Wendlinger

March 2017

1 Introduction

The goal of this project is to provide a tool to detect image tampering realized by inpainting methods, with a focus on robustness to post-inpainting image alterations such as the addition of noise or JPEG compression. The main targets are patch-based algorithms such as the one described in [5], and more generally the broader category identified in [3].

2 Notations

We will use the following notations:

- $\Omega \subseteq \mathbb{Z}$: discrete domain of the image
- $\widetilde{\Omega}$: set of 5×5 patches contained in Ω
- Ω_p is the set of pixels contained in patch $p \in \widetilde{\Omega}$
- $u : \Omega \to \{0, \ldots, 255\}^3$ is the RGB image

Usually patches $p \in \widetilde{\Omega}$ are assimilated to their center pixel, so $\widetilde{\Omega}$ can be seen as a subset of Ω.

3 The detection algorithm

In this section, we describe the algorithm used for detection. Some parameters are introduced, their value is given in section 3.3.

3.1 Preliminary computations

3.1.1 Nearest neighbor field (NNF)

The NNF $\varphi : \widetilde{\Omega} \to \widetilde{\Omega}$ is defined by:

$$\forall p \in \widetilde{\Omega}, \quad \varphi(p) = \arg\min\{d(p, q) \,|\, q \in \widetilde{\Omega}\,,\ \|p - q\| \geq \tau_0\}$$

Where d is a radiometric distance on the image (we used the euclidian distance between the RGB values of the pixels in the patch), and τ_0 is a minimum distance (for pixel coordinates) between a patch and its nearest-neighbor. If not for τ_0 the NNF would trivially be the identity function.

Computing the exact NNF is too expensive to be of any practical use, however a relatively good approximation can be obtained using the PatchMatch algorithm described in [1]. PatchMatch is actually also used by almost all of the patch-based inpainting methods.

3.1.2 Flat patches

In some steps of the algorithm we want to ignore patches that are "too flat", since they create unwanted detections. To do this, we consider a patch p too flat when its variance is smaller than a constant σ^2_{min}.

$$\forall p \in \widetilde{\Omega},\ \mathrm{Var}(p) = \sum_{z \in \Omega_p} \|u(z) - \bar{u}(p)\|^2$$

$\bar{u}(p)$ is the mean value of u on Ω_p.

3.2 Performing the detection

The detection algorithm relies on the fact that patch-based algorithm can be assimilated to several copy-moves of some regions in the images. Indeed, even if this is not done explicitly, patch-based algorithms tend to copy rather large chunks of the image in the inpainted hole.

The inpainting detection is performed in two steps:

- identify the dominant offsets
- check whether each dominant offset is due to inpainting

3.2.1 Identifying dominant offsets

To a patch p we can map an offset $\varphi(p) - p$. In an inpainted image, one can expect that offsets corresponding to the copying of a region of the image are more represented in the image than other offsets, we call them "dominant". To identify them, we simply count the number of times each offset appears in the NNF, and retain the most represented offsets up to a number of $N_{dominant}$. If an offset originates from a patch with a too low variance (3.1.2) it is not counted.

One important advantage of this approach is that it is robust to noise and other alterations of the image after inpainting: while an offset corresponding to a copy-move may appear less consistently in the case of such an alteration, it will appear often enough to be detected.

(a) image with copy move (b) distribution of o (c) noise $\sigma = 5$

Figure 1: Distribution on the image of a dominant offset o corresponding to a large copy-move region

(a) image with copy move (b) distribution of o (c) noise $\sigma = 5$

Figure 2: Distribution on the image of a dominant offset o corresponding to a smaller copy-move region

3.2.2 Validating a dominant offset

Distribution of the offset Each dominant offset goes then through a validation step, to check whether it corresponds to an inpainting operation. In the case of such an operation on a perfect image, one can expect the offset to appear in large connected zones on the set of patches, as can be seen in figures 1b and 2b. However, after alteration of the image such as the addition of noise this is not necessarily true anymore : figures 1c and 1c show that for the second image, the distribution of the offset on the noisy image is not consistent enough for the offset to be identified as originating from an inpainting.

Relevance map That is why we instead chose to build the relevance map of an offset o:

$$\forall p \in \widetilde{\Omega} \quad R_o(p) = \mathbb{1}\{p+o \in \widetilde{\Omega}\} \cdot \mathbb{1}\{d(p, p+o) \leq \alpha \cdot d(p, \varphi(p))\} \cdot \mathbb{1}\{\mathrm{Var}(p) \geq \sigma_{min}^2\}$$

Intuitively, an offset is considered relevant for a patch p if it maps it to a patch "close enough" relatively to its nearest neighbor.

Bidirectional relevance map Since we expect the symmetric of an offset to be relevant, we can build the bidirectional relevance map:

$$\forall p \in \widetilde{\Omega} \quad B_o(p) = R_o(p) \cdot R_{-o}(p+o)$$

Figure 3: Offset relevance for the same images and offsets as in figures 1 and 2 after addition of noise

From now on, whenever we mention the relevance map we will actually mean the bidirectional relevance map. Figure 3 shows the relevance maps for the precedent images after the addition of noise: while some spurious relevances appear, the relevance map for the offset considered presents a much more consistent zone than the distribution of offsets displayed in figure 2.

Validation To mark an offset as corresponding to an inpainting operation, we perform the following operations on the relevance map:

- compute N_{big} the number of pixels in connected zones of more than A_1 pixels
- compute N_{small} the number of pixels in connected zones between A_0 and A_1 pixels
- if we find a connected zone of more than A_2 pixels, we skip the computation and consider the image inpainted
- if $N_{big}/N_{small} \geq r$ the image is considered inpainted

Detection mask We can easily use this detection to create a detection mask for all offsets, by aggregating all the positive detections on the same mask. While having a precise delimitation of the inpainting hole is not the goal of this method, the masks are a useful tool to visualize the results of the algorithm.

False positive on small offsets Experimentation has shown that for non inpainted images, especially after JPEG compression, many offsets appear near the threshold τ_0 defined in 3.1.1. To avoid them, we simply ignore the offsets o where $\|o\| < \tau_1$.

3.2.3 Dealing with autosimilarity

Man made structures such as building and fences tend to create false positives: there are indeed in photographs containing them regions that are so similar that

(a) Untampered image

(b) Relevance map

Figure 4: Example of relevance map in the case of man-made structures

they can seem like the result of an inpainting. This is the reason why we use a rather complex validation of the relevance map using 4 different thresholds instead of a more straightforward one: because experience proves that it deals much better with autosimilarity induced by man made structures.

3.3 The hyperparameters

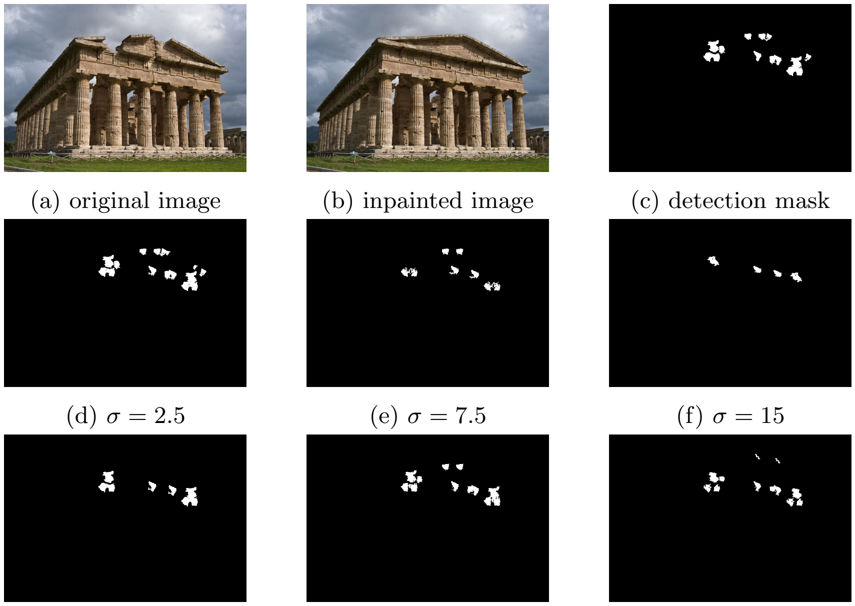
One drawback of this algorithms is that it involves many hyperparameters, whose value can be hard to choose. Here is a list of the hyperparameters and the values we chose:

- $\tau_0 = 20$ minimum distance between a patch and his nearest-neighbor (3.1.1)
- $\tau_1 = 25$ minimum norm of an offset for it to be valid (3.2.2)
- $\sigma^2_{min} = 50$ minimum variance of a patch (3.1.2)
- $N_{dominant} = 20$ the number of dominant offsets (3.2.1)
- $\alpha = 1.14$ used in the computation of the relevance map (3.2.2)
- $A_0 = 5, A_1 = 100, A_2 = 500, r = 5$ used in the validation of the relevance map (3.2.2)

4 Experimental results

We tested our algorithm on:

- images from research papers about patch-based inpainting methods
- inpainted images taken from various places on the internet
- images from the copy-move database features in [2]



(a) original image (b) inpainted image (c) detection mask

(d) $\sigma = 2.5$ (e) $\sigma = 7.5$ (f) $\sigma = 15$

(g) JPEG QF = 80 (h) JPEG QF = 50 (i) JPEG QF = 20

Figure 5: Example of detection with post-inpainting addition of uniform gaussian noise and JPEG compression

4.1 Images from research papers

Since the algorithm is specifically targeted towards inpainting with patch-based methods, performance is really good on images extracted from the papers describing these methods. Figures 5, 6 and 7 show the detection masks on the inpainted image after alteration by addition of a uniform Gaussian noise (the value of the standard deviation is given for color intensities between 0 and 255) or JPEG compression with various quality factors (QF).

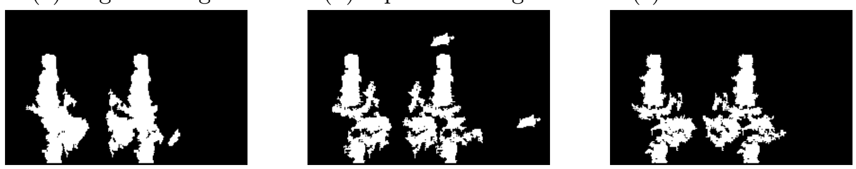
4.2 Images from the Internet

To see how well the algorithm would apply to real world forgeries, we collected images from various places on the Internet, among them the Photoshop Request subreddit [4] where users can request for specific forgeries on their photographs.

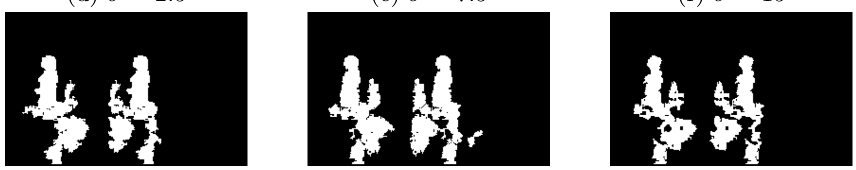
While some forgeries go completely undetected, it turns out that most forgeries, even if they are polished with hand-made adjustments, involve some copy-move operations. Consequently, the algorithm is able to detect them.

Figures 8 and 9 show typical results of detection on image where the technique used for the forgery is unknown: most of the forgery is undetected, but the parts that are copy-move operations are detected. Since the detected regions

(a) original image (b) inpainted image (c) detection mask



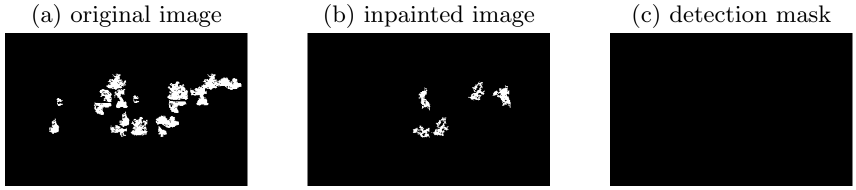
(d) $\sigma = 2.5$ (e) $\sigma = 7.5$ (f) $\sigma = 15$



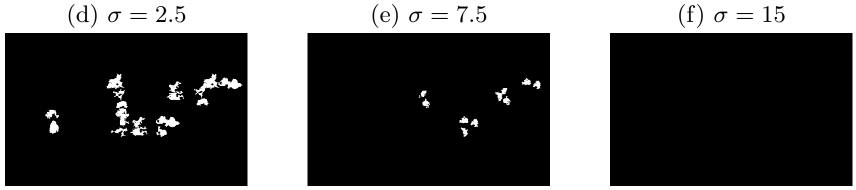
(g) JPEG QF = 80 (h) JPEG QF = 50 (i) JPEG QF = 20

Figure 6: Example of detection with post-inpainting addition of uniform gaussian noise and JPEG compression

(a) original image (b) inpainted image (c) detection mask



(d) $\sigma = 2.5$ (e) $\sigma = 7.5$ (f) $\sigma = 15$



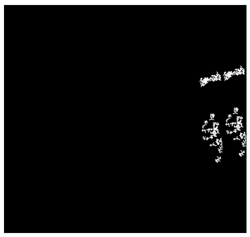
(g) JPEG QF = 80 (h) JPEG QF = 50 (i) JPEG QF = 20

Figure 7: Example of detection with post-inpainting addition of uniform gaussian noise and JPEG compression

(a) original image

(b) inpainted image

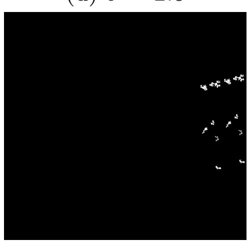
(c) detection mask



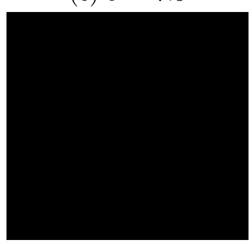
(d) $\sigma = 2.5$

(e) $\sigma = 7.5$

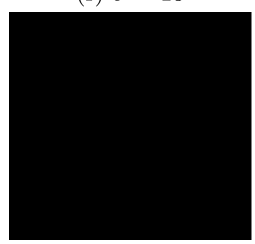
(f) $\sigma = 15$



(g) JPEG QF = 80



(h) JPEG QF = 50



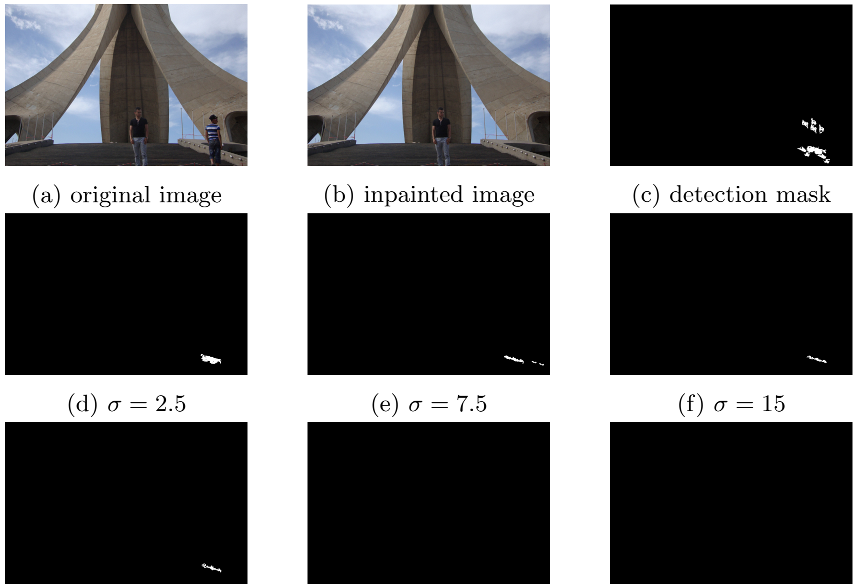
(i) JPEG QF = 20

Figure 8: Example of detection with post-inpainting addition of uniform gaussian noise and JPEG compression

are smaller, the results are less robust to post-inpainting alterations.

4.3 Comparison to [2]

In [2] the authors developed an algorithm with a slightly different scope from ours: it is restricted to detection of a single copy-move operation of a rather large region, but can handle small rotations and rescaling of the copied region. However, the case of detecting a single copy-move without the rotation and scaling is a sub-problem of ours, and since the authors established a dataset and thoroughly presented their results, we can use them as a benchmark for our own algorithm. Because of the differences we mentioned, the goal is not to perform better than [2] at all cost. In particular, the authors know that they are looking for rather large regions corresponding to a single copy-move, an assumption that we can not make. This makes it much easier for them to avoid false positives described in 3.2.3



(a) original image (b) inpainted image (c) detection mask

(d) $\sigma = 2.5$ (e) $\sigma = 7.5$ (f) $\sigma = 15$

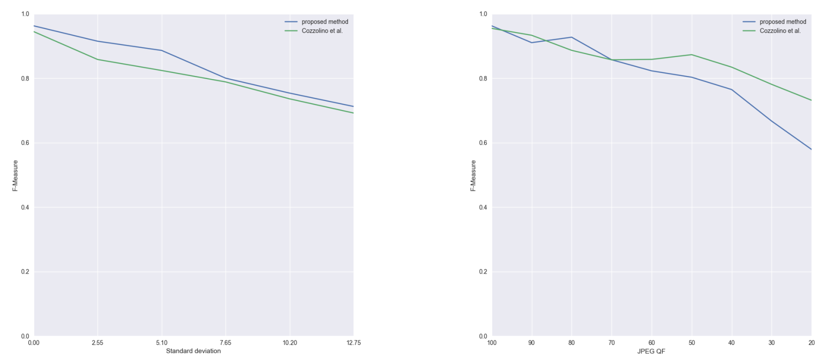
(g) JPEG QF = 80 (h) JPEG QF = 50 (i) JPEG QF = 20

Figure 9: Example of detection with post-inpainting addition of uniform gaussian noise and JPEG compression

Figure 10 shows the comparative results on the F-measure of algorithm at an image level (whether the image is detected as inpainted) for the dataset provided by the authors after addition of noise or JPEG compression. Our algorithm has globally comparable performance, with better results for moderate noise ($\sigma \in [0, 7]$) but higher sensitivity to low JPEG quality factors ($QF \leq 60$).

4.4 Possible improvements

Even though the results are mostly satisfying, there is still room for improvement. First, as we already mentioned earlier, the algorithm has numerous parameters, and finding adequate values can be hard. Also, the performance on images with post-inpainting JPEG compression is not impressive. The place with the most room for improvement is the analysis of the bidirectional relevance map in the validation step (3.2.2): it involves 4 different parameters whose values were found by trial and error (and have no theoretical justification), and while it really reduces false positives in the case of auto-similar man-made structures, it may as well be responsible for the poor performance in presence of JPEG compression.



(a) uniform Gaussian noise

(b) JPEG compression

Figure 10: Comparison with the algorithm developed in [2]

5 Attempted approaches

In this section, we describe other approaches that we attempted. All of them, while rather efficient on perfect images, did not give satisfactory results in the case of post-inpainting image alterations.

5.1 Connected components on the offset map

In the inpainted region, the offset map $p \to \varphi(p) - p$ is expected to be constant. A simple approach can be to apply a threshold on the size of these constant regions in the offset map to detect inpainting.

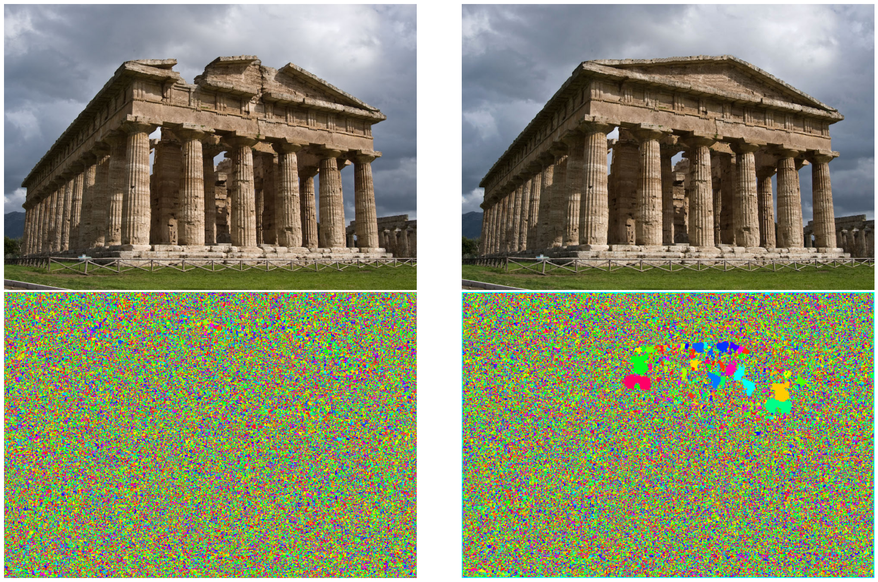
Figure 11 gives an example of the connected components of the offset map.

5.2 Symmetric patches

A symmetric patch p is a patch whose nearest-neighbor's nearest neighbor is itself : $\varphi\left(\varphi(p)\right) = p$. We can create a symmetry map $\mathcal{S} : \widetilde{\Omega} \to \{0, 1\}$ defined by:

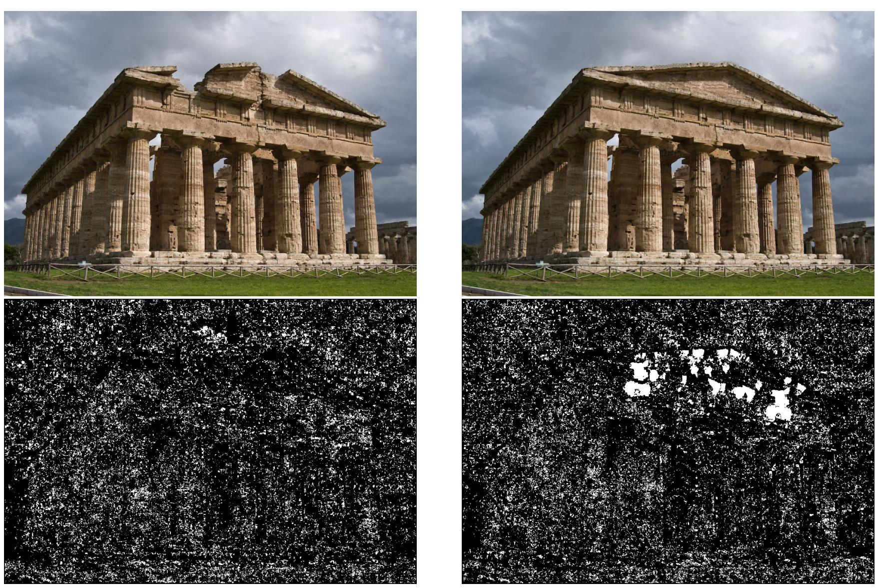
$$\forall p \in \widetilde{\Omega},\ \mathcal{S}(p) = \mathbb{1}_{\varphi(\varphi(p))=p}$$

Figure 12 shows an example of symmetry map for an untampered and an inpainted image. In inpainted images the symmetry map features clusters of positive pixels, while on regular images the symmetry map is rather sparse. Our first approach was actually to perform a morphological opening of the symmetry map and consider the image inpainted if there were still positive pixels after the operation.



(a) original image (b) inpainted image

Figure 11: Connected components of the offset map (each unique component is assigned a random color)



(a) original image (b) inpainted image

Figure 12: Example of symmetry maps

References

[1] Connelly Barnes et al. “PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (Aug. 2009).

[2] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. “Efficient Dense-Field Copy-Move Forgery Detection.” In: *IEEE Trans. Information Forensics and Security* 10.11 (2015), pp. 2284–2297. URL: `http://dblp.uni-trier.de/db/journals/tifs/tifs10.html#CozzolinoPV15`.

[3] Vadim Fedorov, Gabriele Facciolo, and Pablo Arias. “Variational Framework for Non-Local Inpainting”. In: *Image Processing On Line* 5 (2015), pp. 362–386. DOI: `10.5201/ipol.2015.136`.

[4] *The Photoshop Request subreddit.* URL: `https://www.reddit.com/r/PhotoshopRequest/`.

[5] Yonatan Wexler, Eli Shechtman, and Michal Irani. “Space-Time Completion of Video”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.3 (Mar. 2007), pp. 463–476. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2007.60`. URL: `http://dx.doi.org/10.1109/TPAMI.2007.60`.